The Romanian-Latin-Hungarian-German Lexicon - The Lexicon of Buda (1825). Informatics Challenges for an Emended and On-Line Ready Edition[1]

Daniel-Corneliu Leucuta, Bogdan Harhata, Lilla Marta Vremir & Maria Aldea

Keywords: *informatics challenges, multilingual, old lexicon.*

Abstract

The Lexicon of Buda or *the Romanian-Latin-Hungarian-German Lexicon*, published in 1825 in Buda, is the first etymological and explicative, quadrilingual Romanian dictionary. The roughly 13,000 entries / 771 pages lexicon are an important cultural heritage, representing the collective cultural memory of those times, offering a testimony in the life, circulation and evolution of many words. The aim of this paper is to present the informatics challenges in the creation of an emended and on-line ready edition of *the Lexicon of Buda*.

1. Background

The Lexicon of Buda or *the Romanian-Latin-Hungarian-German Lexicon*, published in 1825, is the first etymological and explicative dictionary of the Romanian language. It represents a collective cultural memory of those times, which offers a testimony in the life, circulation and evolution of many words. Beyond its cultural value, this dictionary, written in four languages, marks the beginning of modern Romanian lexicography. Thanks to its apparatus, which aligns it with similar European works, *the Lexicon of Buda* represents an important linguistic and didactic tool. Comprising more than 13,000 entries on 771 pages, the lexicon combines some previous sources with the successive work of many authors over a period of 30 years.

There is already a scanned and pure machine optical character recognized on-line version of this work, but its scan quality is insufficient for many pages, and the character recognition is very poor. That's why we have tried to develop a new online platform so as to make the lexicon available online. There are several informatics issues that this lexicon raises for an electronic edition, due to the old orthography, its multilingual nature, its long introduction, annotation necessities and the lack of systematization, as well as to our desired standards regarding advanced search capacities, bibliometrics, several users working on the project, and securing the system from fraudulent usage.

The aim of this paper is to present the informatics challenges of *the Lexicon of Buda* for the creation of its emended and on-line ready edition.

2. Materials and Methods

The emended and on-line ready lexicon is developed with free tools: PHP (PHP) server side scripting language, MySQL database (About MySQL), XHTML (XHTML 1), CSS (Cascading Style Sheets), DOM (Document Object Model), JavaScript (Javascript), AJAX (AJAX: A New Approach to Web Applications), jQuery (jQuery), web pages being served by an Apache http server (Apache HTTP Server).

This system is designed to offer (a) dynamic web pages of the original scanned, transcribed and transliterated version of the introduction, also translated, and of all dictionary entries, next to the emended version, (b) a simple and an advanced search system for both original and emended versions, (c) a system allowing to build the database content, (d) a database of users having the right to access these resources, (e) backup and restore pages, (f) and a system to log site activity, that can be used for bibliometrics and to identify fraudulent access to the website .

3. Results and Discussion

The emended and on-line ready edition of the lexicon is a work in progress. Several informatics issues that the lexicon has raised for this electronic edition have been dealt with or answers to be implemented in the near future have been found.

3.1. *The old orthography*

3.1.1. *Finding and creating the fonts*. The multilingual dictionary contains old orthography for different languages: Gothic letters with particular traits for the German text, Hungarian orthography, the etymological writing of Romanian, the Cyrillic characters for old Romanian. To deal with these different font problems we have chosen Gentium Plus free font (***Gentium – a typeface for the nations***) that covers all letters and diacritics needed for old Latin writing Romanian, Latin and Hungarian. The German Gothic writing was solved with Leipzig Fraktur free font (***Leipzig Fraktur***). The most difficult endeavor has been the old Cyrillic glyphs. No set of Cyrillic glyphs for that period of time is available, and although there are many Cyrillic fonts for many languages, none contain all letters, diacritics and the correct shape for those glyphs. The solution has been to create the Cyrillic with a font drawing software Fontforge (***FontForge***). For diacritics, and accents, special signs have been drawn to accommodate for lower and uppercase characters. This approach, instead of creating glyphs for every letter – diacritic or accent combination, has been used to minimize the number of glyphs. The accents / diacritics can be combined with any letter, being drawn after the letter has been typed. This new font can be made freely accessible in the public domain.

3.1.2. *Working with the fonts in the web interface*. The number of Cyrillic glyphs is far bigger then the keys on the keyboard. The mapping has been done so that Latin letters on the keyboard may have a corresponding Cyrillic glyph. Remaining Cyrillic glyphs, diacritics and accents, and also Hungarian or etymological Romanian orthography, as well as rare letters have been drawn as tiny buttons in the website text editor allowing their insertion. The dictionary entry text box is in fact a text area that is controlled by a WYSIWYG (what you see is what you get) html Editor – TinyMCE (***TinyMCE – Javascript WYSIWYG Editor***). This script is added to the HTML page and can be extended with buttons or other devices. The buttons are similar to the ones in Microsoft Office toolbars and can perform different functions. In our case, the image on the buttons has been defined as that of the glyphs. The code triggered by the buttons is to insert the corresponding code for the letter in the input field for the dictionary entry. The JavaScript code for it is the following:

ed.addButton('ButtonName', {title : 'Cyrillic – leter name', image : 'ImageFileName.png', onclick : function() {ed.focus(); ed.selection.setContent('ś');}}).

3.1.3. *Making uncommon fonts visible in any web browser*. Web browsers render pages with fonts installed on the computer. The fonts used in our project are not commonly installed in current operating systems. This requires that the fonts be offered by the web site so that the browser may render the page with the correct glyphs. Different browsers are able to work with different font file types to this end. The way to make the majority of them work with these special fonts is by converting the font to different font file formats (that can be perceived by different browsers) with FontForge – .eot (Embedded Open Type File Format), .ttf (TrueType Reference Manual), .svg (Scalable Vector Graphics), .otf (OpenType specification) –, and online tools for this – .otf to woff (Convert otf to Woff, WOFF File Format). Next, we have added the following lines of code in the Cascading Style Sheet of the web page: @font-face {font-family: 'romtipar'; src: url('romtipar.eot'); src: url('romtipar.eot?#iefix') format('embedded-opentype'), url('romtipar.woff') format('woff'), url('romtipar.ttf') format('truetype'), url('romtipar.svg#romtipar') format('svg');} (***How to Use @font-face***). In the above code, *romtipar* is the name of the Cyrillic font for old Romanian writing.

3.2. *The multilingual website*

The dictionary has information in several languages. The interest of the dictionary becomes broader, so a multilingual website is needed, for international users. The content in multiple languages, many of them using different and uncommon fonts, exacts a simple and efficient way to display the information correctly.

3.2.1. *Displaying multiple fonts in the same paragraph*. There are several fonts needed to correctly show glyphs in Cyrillic, Gothic, or Latin written texts for each dictionary entry. The solution to this problem was to set a Latin based font, *Gentium Plus*, for the whole dictionary entry as the default font that would cover the old Romanian, Latin and Hungarian texts. For the Cyrillic and Gothic we have defined two formatting buttons that apply a style with the corresponding font (our newly defined Cyrillic font and Leipzig Fraktur respectively) for any text that is selected. This works in a way similar to that of applying a style to a text in Microsoft Word text editor. When linguists enter the text in the dictionary entry text box, they apply this formatting so as to make sure that pages show the correct font for each piece of text, irrespective of the language. The code for this solution is part of the code for the action triggered by the formatting button in the HTML editor: ed.selection.setContent ('<span class="LeXBuDcyrillic">' + ed.selection.getContent() + '</span>'). Here, *LeXBuDcyrillic* means the style for the Cyrillic text, that in a Cascading Style Sheet (CSS) file is defined to show the old Cyrillic font.

3.2.2. *Printing pages in multiple languages*. When the on-line dictionary is completed, its whole text will also be used to create a printed version of it. In order to correctly render the fonts in a Word document where the text will be copy pasted, we have created a listing version of the dictionary that allows this. The listing version replaces the span class = "LeXBuDcyrillic" defining the style with CSS files that Word won't understand due to unusual style name with a style = “font-family:romtipar;”. This, combined with the font file installed on the printing computer, will allow proper printing of the lexicon content.

3.3. *The lack of systematization*

The dictionary was written over a period of 30 years, by many authors, almost two hundred years ago. This resulted in a lack of systematization of the content structure and numbering system. The project is supposed to create two versions of the lexicon, an identical replica in an electronic format, and an emended one. One possibility was to build a hierarchical system structure for dictionary entries that could cover all the possibilities of content structures and coding systems. This structure would be filled with information and then, when needed, the original content could be recreated programmatically. The advantage of this system is that it would allow a structured content storage of information, very flexible and advanced search functionalities, automatic numbering of voices, definitions, and contextualization. Its disadvantages are due to programming and working with such a hierarchical system input structure, programming the rendering of the original text, and the lack of systematization that would result in an overly complex tree structure. Since in, our approach we have meant to create a text tagging system, pieces of text that have a specific utility (etymology, notes, grammar information, and so on) can be selected by philologists and associated with a certain tag. The tagging system is similar to XML tags, so, before the selected text an <etym> structure, and at its end an </etym> structure is written. Those tags can be handwritten, or they can be applied by clicking a button in the toolbar of the dictionary entry text editor. Easily to understand and to be used by philologists, easy to program, this technique doesn’t result in overly complex structures if the tag system is not overly extended. Such a tagging system can associate several tags to the same piece of text, provided that it does not intersect starting and ending tags. This would also enable the creation of a hierarchical structure, if needed. The tagging system permits advanced search functions, for specific tagged texts, with further programming steps.

3.4. *Simple and advanced search necessities*

The lexicon should allow for simple and advanced search techniques. The simple search already enables searching for title words, and full text search inside the original ancient text as well as in the emended version. The advanced search technique allows for searching specific tags by typing them in the search box. The user friendly interface will offer a radio box list that will facilitate searching for text in specific tagged texts (etymology, notes, grammar etc.).

3.4.1. *Searching for text in specific tagged texts.* Due to the way text is tagged, a search within the field where the entire dictionary entry is held would result in false positive results. That is, if one searches for a text within a tagged text (e.g. *etymology*), when a searched text is found in a dictionary entry, which also has a piece of text with the tag of interest the found text might lie outside the tagged one. To avoid this for any dictionary entry separate fields for each tag are created every time the entry is saved. This allows for a specific search within fields specially designed for given tags. The same technique is applied for searching text written in a certain language.

3.4.2. *Dealing with accents and diacritics.* Accents and diacritics can make the search difficult due to the fact that search engines are designed to find exact matches. Users of search tools expect the results to be with or without accents and diacritics. In order to have diacritic-free versions for all dictionary entries, a separate field is created in the database to hold the accent and diacritics stripped version of the entry. This field will then also be used for the actual search queries.

3.4.3. *Ranking advanced full text search results by relevance*. In order to rank search results by relevance, MySQL full-text search capacities can be used: MATCH, AGAINST, along with Boolean operators such as truncation, negation, exact phrase, sub-expressions etc. (***Getting Started with MySQL's Full-Text Search Capacities***).

3.5. *Bibliometrics*

Having the text tagged for lexical, grammatical, etymological, multilanguage information allows for counting and computing statistics for any tag used in the system. This information can later be analyzed.

3.6. *Multiple users working on the project*

3.6.1. *Preventing simultaneous work.* In order to preserve the database integrity by preventing simultaneous work, each time a user opens a dictionary entry for editing, a field associated to the entry is set to – work in progress, and the id of the user who is working on it. When another user watches the list of entries the entries that are edited at that time are marked as – work in progress. Those marked entries can be opened but with a warning about being careful not to corrupt the work, and a text informing who the user working on that entry is. When an entry is saved and its windows closed the status of the entry changes, so it is not a work in progress anymore. The process is fulfilled with AJAX code.

3.6.2. *Allowing communication.* Problems dealing with dictionary entries can arise any time. If users encounter such a problem, they can check a checkbox that marks the entry as having a problem. Then, the user can describe the problem in a text field. Other users can filter the entries to see what other problems there are, and they can answer in the text field. A note for philologists is also available, to inform the others on specific issues regarding specific entries. Beside this technique, and beside team meetings, the users can use regular instant messaging software to communicate.

3.6.3. *Tracking log of dictionary entry versions.* Working in a team on each dictionary entry, can at times be difficult due to the asynchronous work on the entries. If any user erases a part of the text that was well written, or modifies something another user worked on previously, then problems may arise. To deal with these situations, a tracking log of each entry version is kept. Thus each time an entry is saved, the entry is added to a big field that contains all the previous saved versions of the entry, along with the name of the user who saved them, as well as the date and time stamp. This log can be used to retrieve old good work.

3.7. *Securing the system from fraudulent usage*

Any website should be secured for improper or fraudulent usage. This lexicon website uses a security in depth approach. Several security mechanisms are put in place to validate input, to prevent SQL injection attacks, cross site scripting and other attacks. The most important attack vectors for a dictionary are: the search box, GET and POST requests that can be forged to inject harmful SQL code into the database; the cross site scripting when the search terms are shown on the results web page, if they are not filtered, so that they can place harmful JavaScript code on the web page. To prevent such attacks, all inputs are filtered with Regular Expressions, content rules, length rules, and text transformation filters.

4. Conclusions

The insufficient quality of electronic versions of the dictionary has made us try to develop a new online platform so as to make the lexicon available online. There are informatics issues that this lexicon raises for an electronic edition. Thus, the paper discusses informatics problems due to the old orthography, its multilingual nature, long introduction, annotation necessities and the lack of systematization, as well as challenges regarding advanced search necessities, bibliometrics, several users working on the project, and securing the system from fraudulent usage.

Note

[1] This work has been supported by a grant of the Romanian National Authority for Scientific Research, CNCS - UEFISCDI, project number PN-II-RU-TE-2011-3-0170.

References

A. Dictionaires

***Lesicon romănescu-lătinescu-ungurescu-nemţescu quare de mai mulţi autori, în cursul a trideci, şi mai multoru ani s'au lucrat seu Lexicon valachico-latino-hungarico-germanicum quod a pluribus auctoribus decursu triginta et amplius annorum elaboratum est*. 1825.** Budae: Typis et Sumtibus Typografiae Regiae Universitatis Hungaricae. (The Lexicon of Buda)

B. Other literature

About MySQL. 6 Mars 2012. http://www.mysql.com/about/.

Ajax: A New Approach to Web Applications. 6 Mars 2012. http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications.

Apache HTTP Server. 6 Mars 2012. https://projects.apache.org/projects/http_server.html.
Cascading Style Sheets home page. 6 Mars 2012. http://www.w3.org/Style/CSS/.
Convert otf to woff. 6 Mars 2012. http://orionevent.comxa.com/otf2woff.html.
Document Object Model (DOM). 6 Mars 2012. http://www.w3.org/DOM/.
Embedded OpenType (EOT) File Format. 6 Mars 2012. http://www.w3.org/Submission/2008/SUBM-EOT-20080305/.
FontForge. 6 Mars 2012. http://fontforge.sourceforge.net/.
Gentium — a typeface for the nations. 6 Mars 2012. http://scripts.sil.org/cms/scripts/page.php?item_id=Gentium.
Getting Started With MySQL's Full-Text Search Capabilities. 6 Mars 2012. http://www.devarticles.com/c/a/MySQL/Getting-Started-With-MySQLs-Full-Text-Search-Capabilities/4/.
How to Use @font-face. 6 Mars 2012. http://boldperspective.com/2011/how-to-use-css-font-face/.
Javascript. 6 Mars 2012. https://developer.mozilla.org/en/JavaScript.
jQuery. 6 Mars 2012. http://jquery.com/.
Leipzig Fraktur. 6 Mars 2012. http://www.peter-wiegel.de/Leipzig.html.
OpenType specification. 6 Mars 2012. https://www.microsoft.com/typography/otspec/default.htm.
PHP. 6 Mars 2012. http://php.net/manual/en/introduction.php.
Scalable Vector Graphics (SVG). 6 Mars 2012. http://www.w3.org/Graphics/SVG/.
TinyMCE - Javascript WYSIWYG Editor. 6 Mars 2012. http://www.tinymce.com/.
TrueType Reference Manual. 6 Mars 2012. https://developer.apple.com/fonts/TTRefMan/index.html.
WOFF File Format 1.0. 6 Mars 2012. http://www.w3.org/TR/WOFF/.
XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition). 6 Mars 2012. http://www.w3.org/TR/xhtml1/.